Follow-the-leader Formation Marching Through a Scalable $O(\log_2 n)$ Parallel Architecture.

J. Colorado, A. Barrientos, C. Rossi, J. del Cerro
Robotics and Cybernetics Research Group
Robotics and Automation Center UPM-CSIC, Madrid, Spain
jcolorado@etsii.upm.es, antonio.barrientos@upm.es

Abstract— An important topic in the field of Multi Robot Systems focuses on motion coordination and synchronization for formation keeping. Although several works have addressed such problem, little attention has been devoted to study the computational complexity within the framework of large-scale systems. This paper presents our current work on how to achieve high computational performance for systems composed by a large number of robots that must fulfill with a marching and formation task. A scalable Multi-Processor Parallel Architecture is introduced with the purpose of achieving scalability, i.e., computation time of $O(\log_2 n)$ for a *n*-robots system. Our architecture has been tested onto a multi-processor system and validated against several simulations testing.

***Index Terms* — Cooperative robotics, Distributed systems, Multi-robot systems, Newton-Euler formulations, Strictly parallel computation, Formation keeping, Convoying.**

I. Introduction

Multi-robot systems (MRS) are an active field that offers rich application domains and research topics. A variety of techniques have been proposed in order to approach the problems of cooperation and coordination in different kinds of applications [1], such as exploration and mapping, search and rescue, environmental monitoring, sensor networks, manipulation and transportation, etc. The field of MRS is nowadays quite extended, ranging from swarm robotics, where a high number of usually homogeneous units are involved, to systems composed by few sophisticated robots with different capabilities.

Multi-robot cooperation applications can be roughly divided in two classes: *tight* or *loose*. Tight cooperation requires a continuous coordination between the robots, e.g. box pushing and formation keeping, whereas loose cooperation requires coordination at the beginning of the mission for planning a division of labour, e.g. exploration and mapping.

This paper focuses on the first kind of applications, concretely in formation marching that require robots to form-up and move in a specific pattern. We first approach the problem from a *formation control* perspective, and then we introduce a novel methodology for improving existing formation control methods from a *performance* point of view.

A. Related Work

In behavior and schema-based models [2],[3],[4], robots act according to a set of predefined behavior patterns, activated in response to the task to perform and their perception of the environment (including the actions that team-mates are performing). Such techniques have the advantage of being fully distributed and relatively easy to implement. Nonetheless, since formation is a consequence of the individual behaviors, the behavior of the formation is difficult to forecast and analyze mathematically.

Virtual fields, such as potential fields [5], [7], social fields [8], and navigation functions [6], maintain the formation as a result of the combined attraction/repulsion forces between agents and the environment. Such forces draw a virtual vector field over the scenario, and robots move along field lines according to the task. These methods have the disadvantages that, in order to compute the "forces" acting on an individual, the status of all the team-mates must be known, leading to the need of intense all-to-all communications between the team members (centralized).

Recently, spring/dump models [9]-[11] have been proposed to overcome the rigidity of the original virtual structures approaches [12], [13]. In such models, the robots are connected using virtual joints that allow the structure to be temporarily modified by external forces (e.g. useful for obstacle avoidance), providing a good elasticity and allowing the formation to squeeze in order to pass through a narrow passage. Spring/dump models have the main advantage that can take into account the dynamics and kinematics of the individual robots. However, for a large-scale system, i.e. $n>>$, the computational complexity of this method constrains the system to be unsuitable for real-time purposes.

B. Contribution

In this work, we focus the attention on large-scale Follow-The-Leader formation (see Fig. 1). In terms of formation control, our formulation is similar to the spring/dump model, in which position relationship between the robots is modeled as *virtual joints*, and each robot is represented as a rigid body that is virtually connected with another. The formation is then considered as a chain of connected bodies, and the motion control of each robot is treated as a multibody problem resolved *collectively* and *in a distributed way*, based on the Newton Euler formalism of coupled rigid body dynamics (see Fig. 2). This method has the advantage that takes into account the dynamics and kinematics constraints

of the robots, and is also suited for teams of heterogeneous robots.

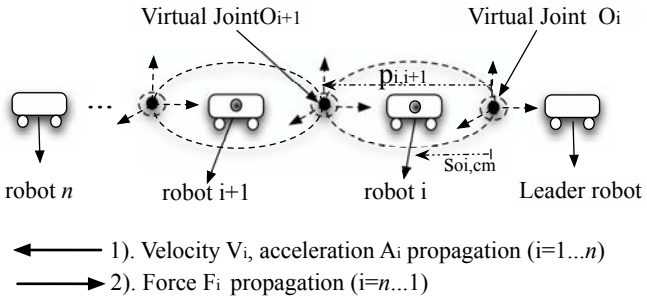


Fig. 1. Follow-The-Leader formation based on the motion presented in serially articulated multibody systems (see Fig. 2).

Another advantage of approaching multi-robot formation based on multibody dynamics is related to locomotion. Considering the dynamics behavior of the system, i.e., internal and external forces, coriolis effects, etc, we are capable of constrain robot formation/marching due to dynamics criteria, e.g., using external forces to avoid obstacles.

On the other hand, we have approached the computational complexity related to the solution of the formation control assignment. The most computationally efficient, and perhaps conventional scheme to describe the problem of motion in serially coupled multibodies, relies on the application of Newton-Euler's set of Equations of Motion (EoM) with an O(*n*) serial complexity (for *n* bodies) [14]. This method is typically used for robotics applications and general multibody dynamics formalisms. Unfortunately, most of the existing O(*n*) algorithms are strictly sequential with bounded parallelism, i.e., performance decreases lineally as *n* increases, and subsequently the scalability of the O(*n*) solutions restrict the system applicability to small-scale scenarios [15],[16].

In this work we take the advances in relation to the scalability of Newtonian formulations [17], in order to re-formulate the classic O(*n*) serial solution of dynamic's EoM into a representation that allow an easy parallelization structure for *computing* and *communicating* the EoM with a computational complexity reduction from O(*n*) to O($\log_2 n$) for an *n*-robot system. In other words, we have a large-scale (*n*>>) MRS where each robot is an independent processing unit capable of *receiving* data, *processing*, and *sending* data[1] to other robots, involving a computation time of O($\log_2 n$).

Using this multibody schema, we are capable of:

- *Implementing large-scale MRS* ensuring system scalability.
- *Addressing virtual dynamics constrains* that allows to consider a wide set of different motion patterns (e.g. force-energy relationships to preserve a specified formation and obstacle avoidance).

Section II reviews the mathematical foundations of classic Newton Euler's multibody-coupled Equations of Motion and also introduces how to improve on the computation of those equations in order to parallelize and distribute them within the MRS. Section III presents the Multi-Processor Parallel Architecture and also describes the parallel computation, communication network, and robot navigation issues. Section IV shows the results from scalability/performance tests and navigation/formation for robot cooperation. Finally, Section V concludes the paper with closing remarks on current and future work.

[1] The word "data" refers to the Equations of Motion –EoM that are being propagated through the Multi-robot System.

II. Distributed Multi-Robot Formation based on Coupled Multibody Dynamics

This section provides the methodology to achieve locomotion based on rigid body dynamics extended to Multi-Robot System. Equations of Motion are presented using spatial algebra operators for improving physical insight. In addition, the use of spatial notation [18] has been very effective in the regard of obtaining high computational efficiency based on the physical variable-compactness.

Next subsection reviews the fundamental concepts of the classic mechanics that are related to the rigid body dynamics modeling, establishing an appropriate mathematical representation of the physical quantities that are involved in that process.

A. Foundations

Assuming from Fig. 2 that the joint frame O_i and the Center of Mass –CM are two points located on the rigid body-*i*, the term $\vec{s}_{oi,cm} \in \Re^3$ is the vector that joints the joint O_i with the CM, and $\vec{p}_{i,i+1} \in \Re^3$ is the vector that joints the joint frames from O_i to O_{i+1}.

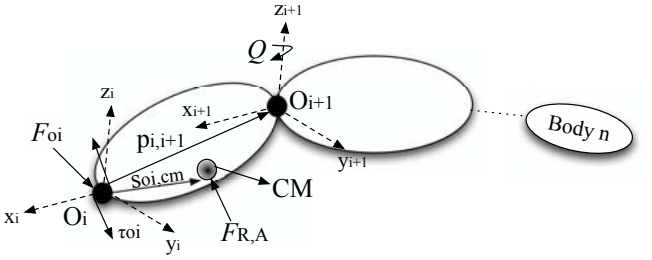


Fig. 2. Coupled rigid multibody description.

The translational and angular velocities $[v,\omega]$ and forces $[f,\tau]$ respectively at any point on a body in $\Re^6$ are:

$$V_{oi} = \begin{bmatrix} \omega_{oi} \\ v_{oi} \end{bmatrix}, \quad \dot{V}_{oi} = \begin{bmatrix} \dot{\omega}_{oi} \\ \dot{v}_{oi} \end{bmatrix}, \quad F_{oi} = \begin{bmatrix} \tau_{oi} \\ f_{oi} \end{bmatrix}. \tag{1}$$

Additionally, in dynamics equations, the spatial quantities in (1) must be propagated and projected onto points or unique frames in order to be operated on. For this purpose, operators for translation $\hat{P}_{i,i+1} \in \Re^{6x6}$ and rotation

$\hat{R}_{i+1,i} \in \Re^{6x6}$ are also defined in spatial operators forms as:

$$\hat{P}_{i,i+1} = \begin{bmatrix} I & \tilde{p}_{i,i+1} \\ 0 & I \end{bmatrix}, \quad R_{i,i+1} = \begin{bmatrix} r_{i+1,i} & 0 \\ 0 & r_{i+1,i} \end{bmatrix}, \qquad (2)$$

where $I \in \Re^{3x3}$ is the identity operator, $\tilde{p}_{i,i+1} \in \Re^{3x3}$ is the skew symmetric matrix corresponding to the vector cross product operator of $\vec{p}_{i,i+1}$. The term $r_{i+1,i} \in \Re^{3x3}$ corresponds to the generalized rotation matrix that takes any point in coordinate frame *i+1* and projects it onto frame *i*. Assuming that the body has a mass m_i and moment of inertia $J_{i,cm}$ about the body's CM, the spatial inertia operator $M_{i,cm}$ in $\Re^6$ is defined as:

$$M_{i,cm} = \begin{bmatrix} J_{i.cm} & 0 \\ 0 & m_i I \end{bmatrix}. \qquad (3)$$

Considering now that the body-*i* in Fig. 2 is serially connected with other bodies, and applying the Newton-Euler foundation based on d'Alambert's principle [19]; the Equations of Motion –EoM are obtained by a:

- *Forward* propagation of velocities V_i in (4) and accelerations $\dot{V}_i$ in (5), from $(i=1)$ to last body $(i=n)$.
- *Backward* propagation of the spatial forces F_i in (6) from last $(i=n)$ to first body in the chain $(i=1)$.

Based on the previous statement, the spatial quantities applied to any body-*i* are composed by the sum of the induced motion (from body *i-1*), and the local component motion (within body-*i*). The spatial velocity in $\Re^6$ is:

$$V_i = \hat{P}^T_{i,i+1} \hat{R}^T_{i+1,i} V_{i-1} + H_i \dot{Q}_i \quad \{\forall_i : i = 1...n\}. \qquad (4)$$

The term $H_i \in \Re^6$ allows the projection of the local velocity component $\dot{Q}_i$ with respect to the axis of motion. Such joints can be for rotation or translation, and Denavit & Hartenberg parameters have been used to define the robots kinematics relations based on homogeneous transformations defined in (2) or quaternion. Differentiating (4), the spatial accelerations are:

$$\dot{V}_i = \hat{P}^T_{i,i+1} \hat{R}^T_{i+1,i} \dot{V}_{i-1} + H_i \ddot{Q}_i + \dot{\hat{P}}^T_{i,i+1} \hat{R}^T_{i+1,i} V_{i-1} + \dot{H}_i \dot{Q}_i \quad \{\forall_i : i = 1...n\}. \qquad (5)$$

The term $\dot{\hat{P}}^T_{i,i+1} \hat{R}^T_{i+1,i} V_{i-1}$ and $\dot{H}_i \dot{Q}_i$ refer to the coriolis and centrifugal accelerations respectively. Finally, the spatial forces are:

$$F_i = M_i \dot{V}_i + \left[\dot{M}_i - \dot{\hat{S}}_{oi,cm} M_i \right] V_i + \hat{R}_{i+1,i} \hat{P}_{i,i+1} F_{i+1} + \hat{S}^T_{oi,cm} F_{R,A} \quad \{\forall_{i:} : i = n...1\}, \qquad (6)$$

where $\hat{S}_{oi,cm} \in \Re^{6x6}$ has the same form expressed by the $\hat{P}_{i,i+1} \in \Re^{6x6}$ operator in (2), and corresponds to the distance between the joint frame O_i and the CM of the body as denoted by the vector $\vec{s}_{oi,cm}$ in Fig. 2. In addition, the term $M_i \dot{V}_i$ is the local force component where $\dot{M}_i V_i - \dot{\hat{S}}_{oi,cm} M_i V_i$ refers to the gyroscopic force effect acting on the body-*i*, and $F_{R,A}$ is any external force acting on body's CM (see Fig. 2).

As previously mentioned, the centralized (*serial*) procedure to compute the EoM in (4), (5), and (6) involves a computational complexity O(*n*). Our goal in next subsection is to reduce that complexity to O($\log_2 n$) by computing the EoM in an efficient distributed procedure.

B. O(log$_2$n) Dynamics to distributed MRS formation

Considering that each robot is a processing unit that composes the MRS. i.e., we have a distributed system with multi-processors capabilities. In order to take advantage of this distributed system, the O(*n*) computation of the EoM is parallelized and efficiently distributed within the MRS. This parallelization and distribution of data is based on the *Forward* and *Backward* schemes previously mentioned.

The solution to this problem involves the reformulation of the EoM from (4) to (6) in a first-order linear inhomogeneous recurrence (LIR) form. This reduces to applying the Kogge and Stone [20] recursive-doubling technique, which reduces the equation set ($\forall_i$), at each one of a total $\log_2 n + 1$ steps, by powers of 2. This procedure allows distributing the EoM computation using *n*-processing units (i.e. *n*-robots), and consequently reduces the complexity from O(*n*) to O($\log_2 n$) in the calculation and propagation of the EoM through the formation scheme. To properly define the LIR reformulation and subsequently the distribution of EoM within the MRS, let focus on the spatial velocity in (4). The goal is to identify how the *i-1* velocity term V_{i-1} affects into the computation of the local term V_i.

$$C_i = \hat{P}^T_{i,i+1} \hat{R}^T_{i+1,i}, \quad B_i = H_i \dot{Q}_i. \qquad (7)$$

Replacing C_i and B_i into (4), and expanding the *i-1* dependent terms until *n* (LIR structure), we obtain:

$$\begin{aligned} V_1 &= C_1 V_0 + B_1, \\ V_2 &= C_2 V_1 + B_2 = C_2 C_1 V_0 + C_2 B_1 + B_2 \\ &\vdots \\ V_n &= C_n V_{n-1} + B_n = C_n C_{n-1} ... C_2 C_1 V_0 + C_n C_{n-1} ... C_2 B_1 + \\ &\quad C_n C_{n-1} ... B_2 + ... + C_n C_{n-1} B_{n-2} + \\ &\quad C_n B_{n-1} + B_n. \end{aligned} \qquad (8)$$

Applying the same procedure for accelerations and forces in (5) and (6), the distribution of the EoM is clearly detailed in Fig. 3. This concept shows how to distribute/propagate the EoM along the robot formation. In Figure 3, the term V corresponds to the velocity or acceleration forward

propagation of (4) and (5) $\forall_{i:}: i = 1...n$, whereas the term F is the force backward propagation of (6) $\forall_{i:}: i = n...1$. Note that, for n=4 robots, just two stages e=2 are required to compute the EoM, i.e., O($\log_2 4$) against O(4). Thus, the system will maintain the scalability to achieve real-time response for increasing number of robots (n>>). Next section shows this process within the framework of MRS.

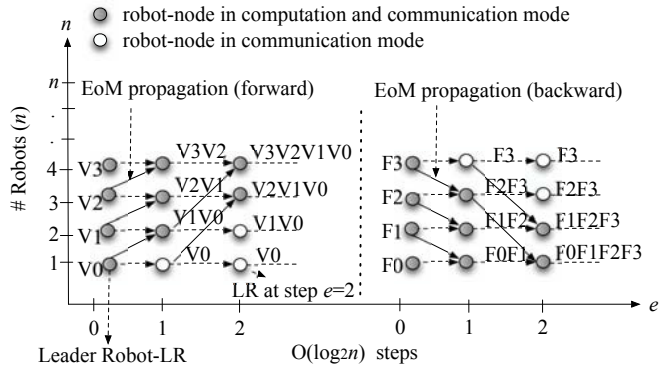


Fig. 3. O($\log_2 n$) propagation scheme of the EoM through the formation *for n=4 robots.*

III. The Multi-Processor Parallel Architecture MP^{2}A

The MP^{2}A is a fully distributed parallel architecture specially conceived for solving formation and marching problems in large-scale MRS. Due to the EoM scheme based on coupled agent behavior, the MP^{2}A supports on the dynamics presented in serial/parallel chains coupled mechanisms, where each robot is capable to move according the physical requirements of the whole system, achieving the best possible synchronization because of the virtually coupling. Three modules basically compose this architecture: *Computation/Propagation*, *Communication*, and *Navigation*. Next subsections review these modules.

A. MP^{2}A Computation/Propagation

This concept was previously introduced in Fig. 3. This section shows how the replacements (LIR EoM structure) in (7) and (8) are applied for computing and propagating velocities, accelerations and forces using the $\log_2 n$ step-approach.

Figure 4 shows the *computation* and *communication* of velocities for n=8 robots with e=3 propagation steps, $\forall_e : e = 0...\log_2 n$. The arrows represent how the velocity EoM flows along the nodes/robots, which each node/robot is represented by a circle. Note that at step e=0, each robot computes their local equations, where the terms B and C are the replacements in (7) respectively. Note that the C term corresponds to a matrix (M) $\in \Re^{6x6}$ whereas the B term is a vector (V) $\in \Re^6$. Furthermore note how the data (matrices and vectors) is propagated and operated in order to compute the total spatial velocities in (4) for each robot-node $\forall_i : i = 1...8$ for a total propagation steps $\forall_e : e = 1...\lceil \log_2 8 \rceil$.

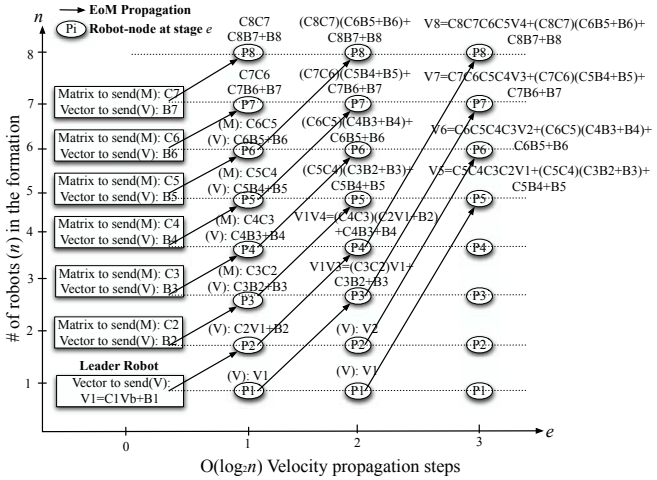


Fig. 4. MP^{2}A O($\log_2 n$) velocity propagation scheme for *n=8 robots*

Additionally from Fig. 4 note that on each step of communication (e), a robot-i sends information to robot $i+2^{e-1}$, and receives information from robot i-2^{e-1} (vice-versa for a backward recurrence). In the case of computing and propagating spatial accelerations (In Fig. 5 consider $A_i = \dot{V}_i$) the C_i and B_i parameters replacements from (5) are:

$$C_i = \hat{P}^T_{i,i+1} \hat{R}^T_{i+1,i}, \quad B_i = H_i \ddot{Q}_i + \dot{\hat{P}}^T_{i,i+1} \hat{R}^T_{i+1,i} V_{i-1} + \dot{H}_i \dot{Q}_i \qquad (9)$$

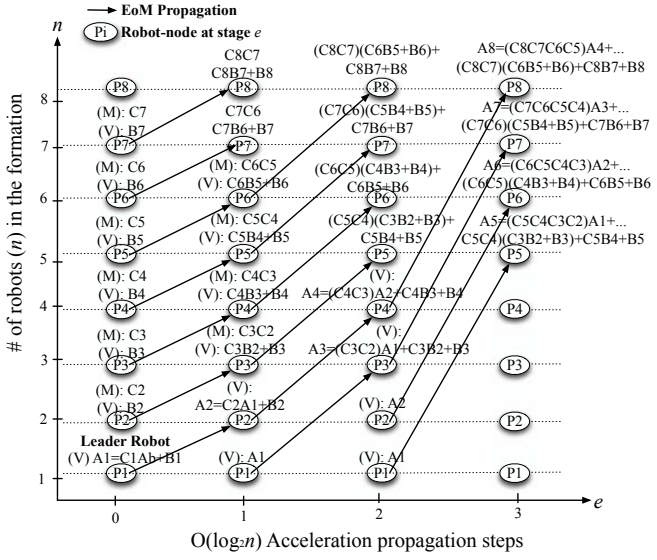


Fig. 5. MP^{2}A O($\log_2 n$) acceleration propagation scheme for *n=8 robots*

Finally, the backward recurrence for computing and propagating spatial forces in (6) are shown in Fig. 6 (the variable F refers to force-EoM). In this case, the C_i, B_i and D_i parameters replacements are:

$$C_i = \hat{R}_{i+1,i} \hat{P}_{i,i+1}, \quad B_i = M_i \dot{V}_i + \left[\dot{M}_i - \dot{\hat{S}}_{oi,cm} M_i \right] V_i, \qquad D_i = \hat{S}^T_{oi,cm} F_{R,A}. \qquad (10)$$

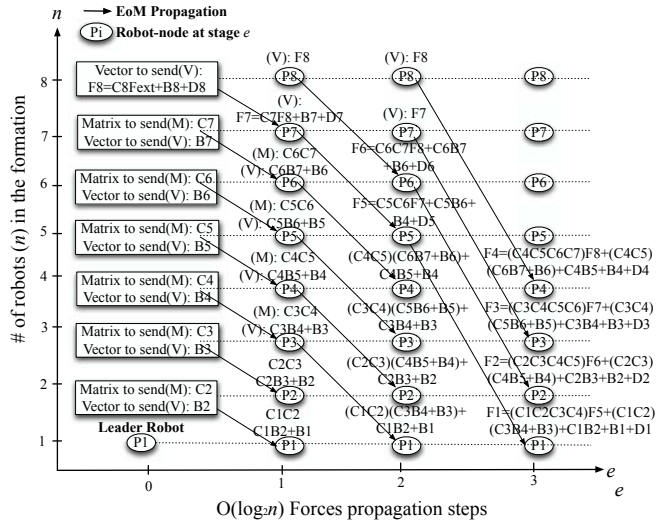


Fig. 6. MP^{2}A O(log$_2n$) force propagation scheme for *n=8 robots.*

Besides the O(log$_2n$) core-distribution of the MP^{2}A, *performance* criterion is also dependent on communication costs. Several factors such as: synchronization, slow communication network, etc, play a significant role in achieving real-time response. Next subsection shows how to include communication issues based on network topology.

B. MP^{2}A network-topology communication costs

Our Follow-The leader formation requires peer-to-peer communication structure in order to send/receive the EoM data. This peer-to-peer system can be implemented in practice using a broken ring network topology, as shown in Fig. 7.

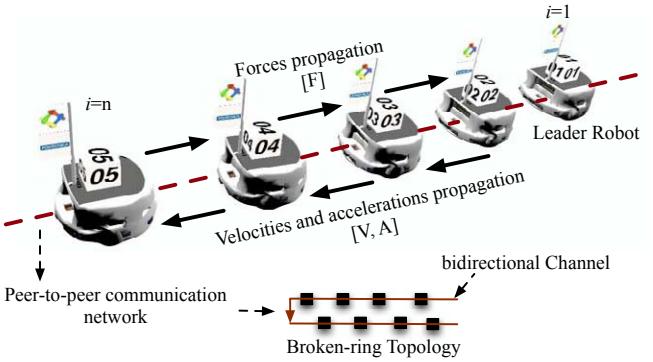
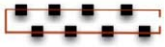


Fig. 7. Peer-to-peer communication structure using a broken-ring network topology.

To measure the total response time of the MP^{2}A, three basic parameters are considered: 1) serial time component (T_s), which is the intrinsic time that each robot delays for receiving information from the previous robot; 2) the parallel time (T_p) that corresponds to the O(log$_2n$) computation time of the EoM showed in Figs 4-6, and 3) the number of robots (n) within the network. Those parameters are related as:

$$T_p = \frac{\lceil \log_2 n \rceil T_s}{n} + L_a + \frac{b_N}{N_{TS}}. \qquad (11)$$

TABLE I
MP^{2}A NETWORK TOPOLOGY MODEL

Topology	Number of vector (V) data
Broken Ring: 	$D_v = \left(2^{\lceil \log 2n \rceil} - 1\right) - \left(2^{\lceil \log 2n \rceil} - n\right)$
	Number of matrix (M) + vector (V) data
	$D_{m\_v} = \sum_{i=1}^{\lceil \log_2 n \rceil - 1} 2^{i-1}$
	Total number of communications
	$D_C = D_v + D_{m\_v}$

The b_N term corresponds to number of transmitted bits, L_a is the latency and N_{TS} the average transmission speed of the network. In order to perform simulation testing of communication costs, we have to develop a specific mathematical model that regards the real number of data that is being transmitted through the network. Using the information related to the number of matrixes (M) and vectors (V) that are propagated in both forward and backward recurrences, we are capable of establishing the total amount of data that is being transmitted within the peer-to-peer structure. This model is shown in Table I.

Using this model, the total number of propagated data D_T (taking into account the three recurrences R_e for computing the EoM; two forward recurrences for velocities and accelerations, and one backward recurrence for forces):

$$D_T = R_e\left(A_b D_{m\_v} + B_b D_v\right) + B_b(n-1), \qquad (12)$$

where A_b is the size of the buffer in the case of sending a matrix (M) plus a vector (V), and the parameter B_b is the size of the buffer is the case of sending just one vector (V). Finally, using (11), (12) and the model in Table I, the total *computation* and *communication* time (T) of the MP^{2}A for n-robots performing a Follow-The-Leader formation task is:

$$T = \frac{\lceil \log_2 n \rceil T_s}{n} + p_t\left[L_a R_e D_C + \frac{2}{N_{TS}}\begin{bmatrix} R_e 42 D_{m\_v} + \\ 6 D_v R_e + 6(n-1) \end{bmatrix}\right]. \qquad (13)$$

Where the p_t term refers to the number of sampled points of the desired trajectory.

C. The MP^{2}A in robot navigation: obstacle avoidance

Obstacle avoidance is an indispensable feature in relation to navigation. Our dynamic model is capable of including an external force command that can be used to drive the robot to the desired direction. This force is applied to the center of mass of any robot within the virtual kinematics chain. This external force is integrated into the Force EoM in (6) as: $\hat{S}^T_{oi,cm} F_{R,A}$. The $\hat{S}^T_{oi,cm}$ operator allows projecting this external force from the CM to the joint frame O_i.

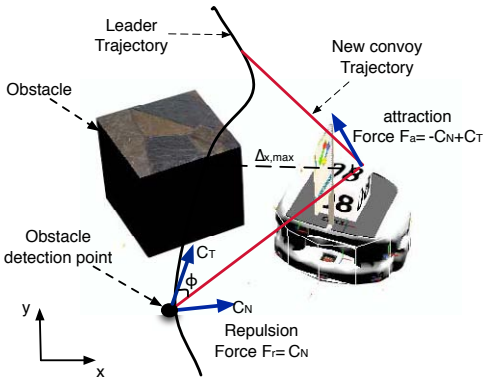


Fig. 8. Obstacle avoidance based on repulsion and attraction external Forces.

We assume the robot is equipped with some kind of sensor (e.g. laser, ultrasonic) capable of detecting the obstacle. Based on this information, the MP^{2}A generates repulsion forces that drive the robot away from the obstacle, and attraction forces that recover the original trajectory path. The obstacle avoidance concept is shown in Fig. 8.

From Figure 8, C_T and C_N are the tangential and normal force vectors used to generate the repulsive force F_r and attraction force F_a. Equation (14) shows the influence of these forces within the motion formation. The term $\Delta_{\max}$ is the orthogonal distance from the robot to the straight line that holds the segment. Likewise, k_R, k_A are the parameters for tuning both repulsive and attraction forces.

$$F_{R,A} = \begin{bmatrix} c\phi & -s\phi \\ s\phi & c\phi \end{bmatrix} \begin{bmatrix} C_N & 0 \\ 0 & -C_N + C_T \end{bmatrix} \begin{bmatrix} c\phi & s\phi \\ -s\phi & c\phi \end{bmatrix} \begin{bmatrix} k_R \\ k_A \Delta^2_{\max} \end{bmatrix} \quad (14)$$

IV. Results

In this section we analyze the performance in terms of cooperative locomotion and scalability of the MP^{2}A within the framework of Follow-The-Leader formation keeping. The theoretical foundations described earlier have been tested with extensive simulations with the aim of:

- Demonstrate the MP^{2}A scalability for large-scale applications.
- Demonstrate reliable robot navigation.

The MP^{2}A algorithms have been coded using C++ programming language. In order to simulate a large-scale robot system, we use an Intel® Core™ 2 Quad processor Q8200 cluster with 4-cores and 8-proccesing threads. Using the multi-core capability of this platform we can use the processors/process that emulates the behavior of a robot. In other words, each processing unit is a robot that composes the MRS. We adopted the MPI v2.0 libraries as a message-passing protocol. Computation and communication times are measured based on the network model in (13).

A. Performance results for large-scale MRS

We have compared the total computation and communication time (T) using our distributed O($\log_2 n$) structure against a centralized, non-distributed O(n) approach. A large-scale MRS with up to n=512 robots that must keep a typical path formation trajectory composed by $p_t = 2000$ sample-points has been considered. Figure 9 shows the time response of the MP^{2}A as a function of increasing the number of robots from n=0 to n=512. Key time-points have been measured for n=64, 128, 256 and 512. Results confirm that our hypothesis of achieving scalability and real-time response when n is larger is been aimed. Besides, the O(n) centralized computation is capable of real-time response for n<32 robots. For a MRS with more than 32 robots, our O($\log_2 n$) approach appear to be suitable.

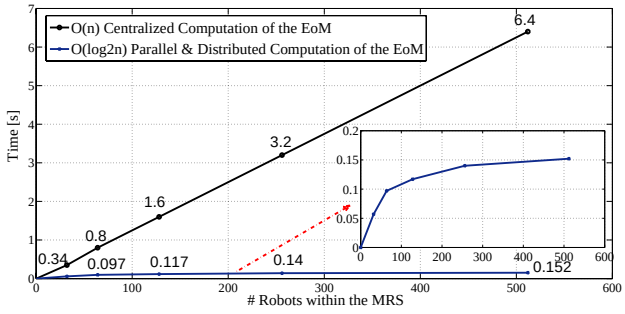


Fig. 9. Testing the O($\log_2$n) distributed MP^{2}A ***VS*** the O(n) centralized architecture, when the number of agents in the MRS is being dramatically increased (execution for *n=512* robots). Numerical results from: Intel® Core™ 2 Quad processor Q8200 cluster.

The time response showed in Fig. 9 corresponds to the computation and communication times for calculating the three EoM propagation recurrences: velocity, acceleration and forces, per each trajectory sample. For n=512 the MP^{2}A demands only 152 milliseconds compared to 6.4 seconds if a non-distributed architecture is used for the same simulation.

For this scenario, the MP^{2}A is about 42-times faster than a centralized computation and propagation of the EoM. Several simulations were conducted varying the number of robots with more complex trajectories with challenging robot maneuvering were also tested (see next subsection for robot navigation results).

Data results from those simulations were used for measuring performance in terms of speedup metric. Speedup provides information of performance as a function of both serial T_s and parallel T_p times and the degree of parallelism of the architecture f_p / n, where f_p is the percentage of parallel work in relation to number of robots n.

$$S_W = \frac{1}{f_s + \frac{f_p}{n}} = \frac{n}{f_s n + f_p}. \quad (15)$$

Where S_W term is the speedup (Amdahl's law) and f_s is the percentage of serial work. In general, speedup is limited by the amount of serial work. As $f_s \Rightarrow 0$ and $f_p \Rightarrow 1$, the result is an ideal speedup equal to the number of available processing units $S_W = n$.

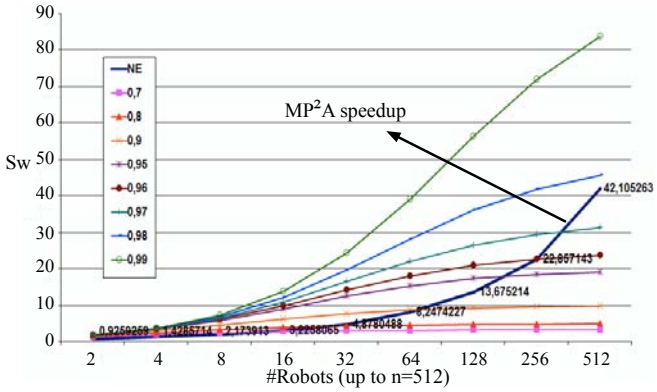


Fig. 10. MP²A speedup for various degrees of parallelism up to *n=512*.

Figure 10 shows the speedup of the MP²A (dark plot) compared to the theoretical speedup achievable for various degrees of parallelism, from 0.7 to 0.99[2] (colored curves). Note that speedup increases sharply after *n*=32. As expected, for a small number of robots/agents the communication overcost compared to computation demanding time has a significant impact on the speedup. Note however that the MP²A benefits are remarkable for *n*>32, with a maximum degree of parallelism of 0.97% in approximately *n*=256.

The maximum degree of parallelism of 0.99% (i.e., best performance) is theoretically achieved by a speedup of $S_W = 83$ for *n*=512. However, communication costs reduces the performance of the architecture almost at a half, achieving a speedup of $S_W = 42.10$ for *n*=512. Despite this, note that results consigned in Fig. 9 (time response) are quite satisfactory.

B. Navigation results

In the first test we used *n*=8 robots that must perform formation keeping with obstacle avoidance capabilities.

The desired trajectory is only known by the leader-robot (head of the formation). Using our EoM propagation scheme, the desired motion is transmitted along the formation. Note that any robot is capable of modifying the original trajectory and transmits such information to the other robots behind. This scheme is very useful for navigation purposes in the case of avoiding obstacles.

Figure 11 shows an example of this experiment. The leader robot has a predefined trajectory that does not consider the obstacles along the path. When the leader detects and avoids the obstacle, the new trajectory information is propagated along the chain, and all the following robots will follow the modified trajectory.

On the other hand, Fig.12 shows the real-time propagation of data within the cluster platform for the experiment in Fig. 11. Packages exchanged among the robots during execution are visualized using the MPI logs of the MP²A running on the cluster platform. Jumpshot profiler was used to observe the O(log$_2$*n*) computation/communication cycles of both backward and forward propagation schemes for computing the EoM and compared them against theoretical propagation scheme showed in Fig. 4-6.

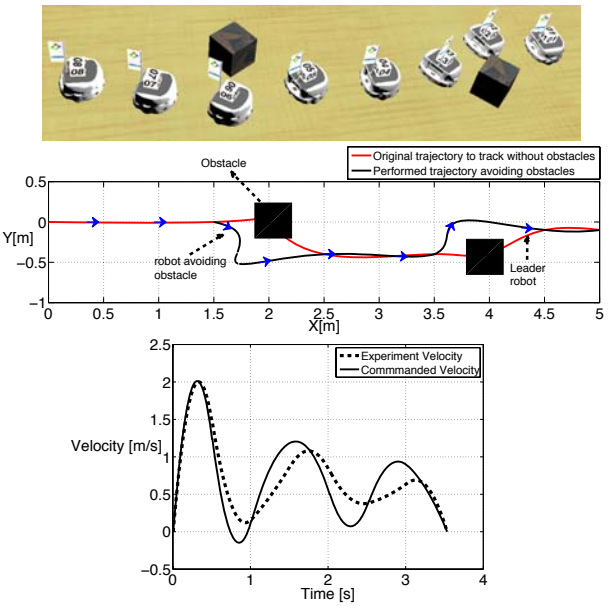


Fig. 11. Robots performing Follow-The-Leader tasks using the MP²A for navigation and obstacle avoidance, *n*=8.

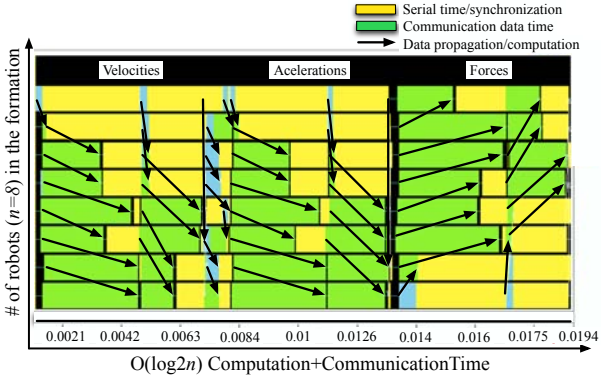


Fig. 12. Execution/communication structure, *n=8* (real time snapshot). Arrows indicate information flow, green areas represent communication time (T_p), and yellow areas are intrinsic synchronization time (T_s) (cf. Eq. (13)).

From Fig. 12 it is clear that spatial velocities has the highest cost due to the required multibody parameters computed in (2) and (3). Velocities require about the 40% of total computation time (*T*=0.0194s).

In the second experiment (Fig. 13) a more complex rigid formation is demonstrated. Three robots aligned in a triangle-shape must maintain such formation in order to transport a box. The purpose of this test is to demonstrate that the MP²A architecture can be used for the motion control of complex configurations, besides the serial chain formation used for Follow-The-Leader in Fig. 11. In this case the communication chain remains serial (peer-to peer system), however the kinematics structure can adapt any serial/parallel chain.

[2] Such range of values is the one typically used for performance evaluation in parallel algorithms based on Amdahl's law.

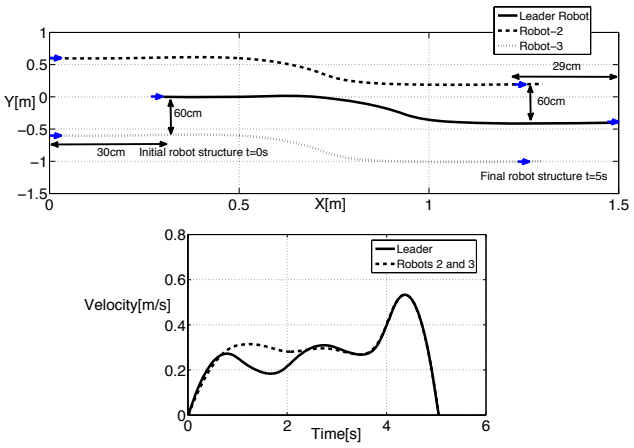


Fig. 13. Robots performing Formation-Keeping task using a non-serial kinematic structure (triangle-shape).

V. Final remarks and Conclusion

Serially rigid-body coupled dynamics methodology was adopted in order to approach Follow-The-Leader robot formation and marching problems in large-scale Multi-Robot Systems. In terms of robot navigation, virtual joints were used to constrain the robots to adopt different formation tasks with high-level of synchronization. The coordinated motion among robots was based on propagating, through the formation, the dynamic's Equation of Motion presented in coupled multibody mechanisms. The computation/communication scheme adopted allowed the MRS to maintain formation and to include external forces in order to avoid obstacles. Several testing using Follow-The-Leader marching were performed, demonstrating the navigation capability of the MP^{2}A.

In terms of performance, the MP^{2}A takes advantage of the available hardware resources to achieve scalability when the number of robots increases. The simulations carried out confirmed that our distributed approach is about *1.2x (*for *n=8)* and *42x (*for *n=512)* times faster compared to centralized computing of the EoM. Current and future work is oriented towards the experimentation of our methodology on a real system.

Acknowledgments

This work is funded by the project FRACTAL, Spain Ministry of Education (DPI 2006-03444), and ROBOCITY 2030 (S-0505/DPI/000235).

References

[1] Parker, L. Current research in multirobot systems. *Artificial Life and Robotics* , *7* (1), 1-1. 2003

[2] Balch, T., and Arkin, R. Motor schema-based formation control for multiagent robot teams. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)* , 1-7. 1995.

[3] Parker, L. ALLIANCE- An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation , 14* (2), 220-240. 1998

[4] Balch, T., and Arkin, R. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation , 14* (6), 926-939. 1998.

[5] Leonard, N., and Fiorelli, E.. Virtual leaders, artificial potentials and coordinated control of groups. *Proceedings of the 40th IEEE Conference on Decision and Control, 2001,3*, 1-7. 2001.

[6] Tanner, H., and Kumar, A. Formation stabilization of multiple agents using decentralized navigation functions. *Robotics: Science and Systems* , 49-56. 2005.

[7] Fazenda, P. Non-Holonomic Robot Formations with Obstacle Compliant Geometry. *PhD Thesis, Universidade Tecnica de Lisboa* , 1-106. 2007.

[8] Reif, J., and Wang, H. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems , 27* (3), 171-194. 1999.

[9] MacArthur, E., & Crane, C. Compliant Formation Control of a Multi-Vehicle System. *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation* , 479-484. 2001.

[10] Gulec, N., and Unel, M. A novel coordination scheme applied to nonholonomic mobile robots. *44th IEEE Conference on Decision and Control,. CDC-ECC'05* , 5089-5094, 2005.

[11] Loizou, S., and Kyriakopoulos. Navigation of multiple kinematically constrained robots. *IEEE Transactions on Robotics , 24* (1), 221-231. 2008.

[12] Lewis, M., and Tan, K. High precision formation control of mobile robots using virtual structures. *Autonomous Robots , 4* (4), 387-403. 2007.

[13] Beard, R., Lawton, J., and Hadaegh, F. A feedback architecture for formation control. *American Control Conference* , 1-33. 2000.

[14] Stepanenko, Y., and Vukobratovic, M. Dynamics of Articulated Open-chain Active Mechanisms. *Math Biosciences , 28*, 137-170. 1976.

[15] Orin, D., McGhee, R., Vukobratovic, M., and Hartoch, G. Kinematic and Kinetic Analysis of Open-chain Linkages Utilizing Newton-Euler Methods. *Mathematical Biosciences* , *43*, 107-130. 1979.

[16] Featherstone, R., and Fijany, A. A technique for analyzing constrained rigid-body systems and its application to the constraint force algorithm. *IEEE Transactions on Robotics and Automation , 15* (6), 1140-1144. 1999.

[17] Featherstone, R. A Divide-and-Conquer Articulated-Body Algorithm for Parallel O(log(n)) Calculation of Rigid-Body Dynamics. Part 1: Basic Algorithm. *The International Journal of Robotics Research , 18* (9), 867-875. 1999.

[18] G. Rodriguez, A. Jain and K. Kreutz-Delgado, "A Spatial Operator Algebra for Manipulator Modelling and Control," *Int. J. Robotics Research*, vol. 10, no. 4, pp. 371-381, 1991

[19] Lanczos, C. The variational principles of mechanics. *4rth Edition ed New York: Dover Publications Inc* , 1-92. 1970.

[20] Kogge, and Stone, P. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. *IEEE Transactions on Computers , 22* (8), 783-791. 1973.